Proactive Caching Placement for Arbitrary Topology with Multi-Hop Forwarding in ICN

Siyang Shan, *Student Member, IEEE,* Chunyan Feng, *Senior Member, IEEE,* Tiankui Zhang, *Senior Member, IEEE,* and Jonathan Loo, *Member, IEEE*

***Abstract*—With the rapid growth of network traffic and the enhancement of the quality of experiences of users, Information-Centric Networking (ICN), which is a content-centric network architecture with named data caching and routing, is proposed to improve the multimedia content distribution efficiency. In arbitrary topology, cache nodes and users are randomly distributed and connected, hence it is challenging to achieve an optimal caching placement under this situation. In this paper, we propose a caching placement algorithm for arbitrary topology in ICN. We formulate an optimization problem of proactive caching placement for arbitrary topology combined with multi-hop forwarding, with an objective to optimize the user delay and the load balancing level of the nodes simultaneously. Since the original problem is NP-hard, we solve the formulated caching placement problem in two sub-problems, content replica allocation sub-problem and content replica placement sub-problem. First, in the content replica allocation sub-problem, the replica number of each content is obtained by utilizing the auction theory. Second, the replica number of each content is used as a constraint for the content replica placement sub-problem, which is solved by matching theory. The caching placement algorithm combined with multi-hop NRR forwarding maximizes the utilization of cache resources in order to achieve better caching performance. The numerical results show that significant hop count savings and load balancing level improvement are attainable via the proposed algorithm.**

***Index Terms*—ICN, IoT, Caching Placement, Multi-hop Forwarding, Arbitrary Topology.**

I. Introduction

RECENT traffic measurements have clearly shown that IP video traffic will account for 82 percent of the overall Internet traffic by 2022, as illustrated in the Cisco VNI forecasts [1]. However, due to the host-to-host communication patterns in traditional IP networks, Internet bandwidth is wasted by repeated transmission of popular contents. In order to solve the problem of low transmission efficiency of content distribution in traditional IP network, ICN has been widely studied [2]. The ubiquitous in-network caching is a key technology of ICN, which avoids the repeated transmission of popular contents and improves the transmission efficiency. In ICN, how to optimize the caching placement and content forwarding to improve the caching performance under limited cache space are one of the main research topics in ICN caching.

S. Shan, C. Feng and T. Zhang are with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China (email:{syshan, cyfeng, tkzhang}@bupt.edu.cn).

J. Loo is with the School of Computing and Engineering, University of West London, London W5 5RF, U.K. (e-mail: jonathan.loo@uwl.ac.uk).

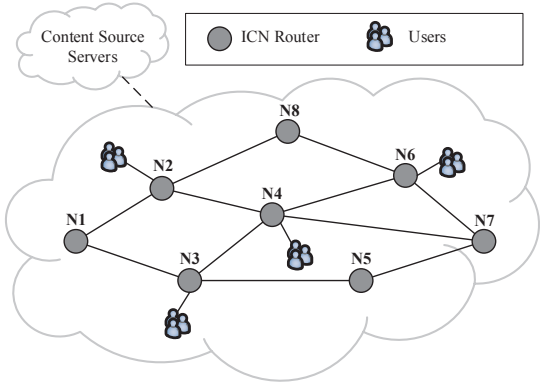


Fig. 1: An example of arbitrary topology

In ICN, the caching placement strategies can be divided into two categories: the strategies for regular topologies and strategies for arbitrary topology [3]. Regular topologies include string topology, grid topology and hierarchical tree topology. In regular topologies, the structures have strong regularity, and thus the content placement and coordination between cache nodes can be determined by solving the analytical model established by prior traffic demand. Most of existing caching placement strategies are regular topology oriented. Arbitrary topology is extracted from the Internet infrastructure [3], and an simple structure of arbitrary topology is presented in Fig. 1. In contrast to the regular topologies, cache nodes and users are randomly distributed and connected in arbitrary topology, which makes explicit collaboration more difficult to achieve [4].

In regular topologies, the position of edge nodes and source nodes are fixed, so the parent-child relationship between the nodes is also fixed. The forwarding path for each edge node is unique, and the explicit collaboration for each edge node is among the cache nodes in a unique forwarding path. However, in arbitrary topology, due to the irregularity of the node connections and the random positions of the edge nodes and the source nodes, the parent-child relationship in arbitrary topology no longer exists. The explicit collaboration for each edge node is not limited to a unique path, and collaboration range for an edge node is enlarged and may be extended to the entire network. Therefore, arbitrary topology is more difficult to perform explicit collaboration.

For research in ICN, [5]–[11] studied the caching placement strategies for arbitrary topology. [5], [6] designed caching placement strategies for arbitrary topology combined with on-

path request forwarding, which considered content popularity and user preferences. The forwarding range of the two strategies is limited due to the on-path forwarding strategy, and thus the caching gain of the cached contents is not fully utilized. [7]–[11] designed caching placement strategies combined with off-path request forwarding. In [7]–[9], cluster head nodes made caching placement decisions to place the contents in nodes inside the cluster, and made forward decisions when receiving content requests. The strategies extended the forwarding range of a request to the current cluster or adjacent clusters, and adopted heuristic strategies to solve the problem, but the forwarding range of content requests was still confined, which limited the improvement of caching gain. In [10], caching placement decisions were made through hash algorithm to place the contents in the cache nodes across the entire networks. When requesting a content, the position of the content replica is calculated by the algorithm. In this strategy, each content has one replica at most, namely single replica caching, and the caching position of the replica is randomly selected without taking content popularity into consideration, which may lead to reduction of caching performances, for caching more popular content replicas can obviously bring more benefits to lower user delay than caching more unpopular contents. According to the above research, due to the limited forwarding range of the content requests, the caching placement strategies for arbitrary topology are incapable of achieving global optimization of the caching performance. [11] determined the content caching locations and forwarding path by solving the optimization problem about routing cost and caching cost. But when forwarding, the cache resources can not be utilized by all users, and one user only requests for one content, which makes the performance gain of the strategy is limited in actual systems.

In summary, multi-hop forwarding is not fully realized in the above researches for arbitrary topology, and we need to study the optimal caching placement problem for arbitrary topology that can achieve global performance optimization. Existing studies indicate that the performance of the caching algorithm for arbitrary topology can be improved by combining with Nearest Replica Routing (NRR) request forwarding strategy [12]–[14]. Multi-hop NRR forwarding is the full version of NRR forwarding strategy, which extends the forwarding range of user requests to any cache node in the network, so that the cache resources could be utilized to the maximum extent. Therefore, in order to decrease cache redundancy, reduce cache replacement, and maximize the utilization of cache resources, we propose a proactive caching placement algorithm combined with multi-hop NRR forwarding for arbitrary topology, to achieve global optimization of the caching performance under arbitrary topology. The innovations and contributions of this work are summarized as follows:

- We proposed a content distribution architecture for arbitrary topology in ICN, where a caching placement optimization problem combined with the multi-hop NRR forwarding is formulated, with an objective to optimize user delay and the load balancing level of the nodes simultaneously. The objective takes into account the user delay and load balancing level of the cache nodes, and makes a trade-off between the two metrics.
- Since the original problem is NP-hard, we solved the formulated caching placement problem in two sub-problems, content replica allocation sub-problem and content replica placement sub-problem. First, we reformulated the original problem into a content replica allocation problem by approximating the optimization objective, from which, the replica number of each content is obtained by utilizing the auction theory to solve the content replica allocation sub-problem. Second, the replica number of each content is taken as a constraint for the content replica placement sub-problem, which is solved by matching theory. And finally, the forwarding routes for each user are determined from the result of the content replica placement problem.
- We evaluated the performance of the proposed algorithm by comparing it with several classical caching algorithms in the case of combining with NRR forwarding algorithm by simulation. The results showed that the proposed algorithm is significantly better than other algorithms, and can effectively reduce the average hop count for content acquisition and improve the load balancing level.

The rest of this paper is arranged as follows: Section II introduces related works in this field. Section III gives the mathematical model of our cache network. Section III-D and IV give the modeling and solution of the content replica allocation problem and the placement problem respectively. The performance verification and comparison of the proposed caching algorithm are given in Section V. And finally, conclusions are summarized in Section VI.

II. Related Works

Currently, lots of researches are studying caching placement strategies based on the on-path forwarding strategy in regular topology [15], [16]. Our research focus, on contrary, on the caching placement strategy based on the off-path forwarding in arbitrary topology. In ICN, the performance of caching is determined by both the caching placement strategy and the request forwarding strategy [17], [18]. The caching placement strategy determines where to cache and what to cache, including a proactive way and a reactive way; and the request forwarding strategy determines the next-hop node when forwarding a user request.

On one hand, depending on the target topology, the caching placement strategies can be divided into strategies for regular topology and caching strategies for arbitrary topology. On the other hand, depending on the forwarding range, the request forwarding strategies can be divided into on-path forwarding strategies and off-path forwarding strategies. The forwarding range of on-path forwarding strategy is confined within the nodes on the default path decided by the default routing strategy, that is, Shortest Path Routing (SPR). The off-path strategy extends the forwarding range beyond the the default path.

Generally speaking, the caching strategies combined with on-path forwarding strategies usually are caching decision strategies, which decide whether to cache the incoming con-

tents on en-route routers or not. While the caching strategies combined with off-path forwarding strategies usually are caching placement strategies, which push contents to the routers proactively.

For instance, [19]–[23] studied the caching strategies for arbitrary topology, where [19]–[21] adopted on-path forwarding strategies, and [22], [23] adopted off-path forwarding strategies. [5]–[11] studied the caching strategies for arbitrary topology, where [5], [6] adopted on-path forwarding strategies, and [7]–[11] adopted off-path forwarding strategies.

Among the caching strategies for arbitrary topology, [5], [6] proposed cache decision strategies based on user preferences and content popularity. [7], [8] proposed caching placement strategies utilizing clustering, which both considered the factors related to topology like neighbor node degree, node centrality. [9] proposed caching placement strategies for multi-domain networks, which utilized the potential values of nodes to decide to cache which contents on which nodes. [10] proposed a caching placement strategy based on the hash indexing of the contents. [11] proposed a joint caching placement and routing strategy utilizing an optimization problem, where several kinds of costs are considered, but the setting of user request probability is too simple that one user only request for one content, which is not practical in the actual systems.

The caching performance of the above strategies is confined due to the limited forwarding range. Several papers have demonstrated the performance advantages of the NRR forwarding strategy among the off-path forwarding strategies [12]–[14]. Adopting NRR forwarding strategy can increase the opportunity for the user requests to be satisfied outside the default path. Users can obtain some contents from nearer routers other than content source servers, so as to reduce the hop count for acquiring contents, decrease the load of the content source servers and improve the utilization of caches.

The existing caching strategies combined with NRR forwarding strategy [11], [24]–[28] are described in detail. [24] studied the joint problem of request forwarding and content caching in cluster-based heterogeneous networks. The optimization objective is to minimize the average content access delay, and an approximate method is used to solve the optimal problem. Content caching and scheduling in cellular networks is studied in [25], which is essentially a joint problem of caching and forwarding. Optimization objectives related to finite queue lengths for elastic traffic and zero average deficit value for the inelastic traffic are given and solved by Lyapunov drift algorithm. [26] studies the caching placement and forwarding problem in ISP networks, as well as the optimization for cache space allocation. Two optimization objectives of minimizing link cost and maximizing cache hit rate are set, and greedy algorithms are used to solve them. [27] studied the problem of caching placement and forwarding in cellular networks. The optimization objective is to minimize the total cost, and an on-line algorithm is used to solve the problem. [28] studied the joint problem of caching and forwarding in a single administrative domain network. The optimization objective is to minimize the maximum link utilization, which is solved by primitive dual decomposition and Lagrange relaxation. The problem of caching placement and forwarding in arbitrary topology is studied in [11]. The optimization objective is to minimize the network traffic, and the heuristic algorithm is used to solve the problem. The research scenarios in [24]–[27] include both wired and wireless networks. Although NRR strategy is realized by means of setting a restricted network topology, transforming the network into a bipartite graph by clustering or partitioning, or simply adopting single-hop wireless access network, all of them stay at the level of single-hop forwarding. The optimization problem with single-hop forwarding is simplified to linear programming problem when the objective is to minimize routing cost. In these schemes, the requesting node is directly connected to the cache in a single hop, and none of these schemes can be generalized to multi-hop scenarios because it results in a non-convex property [29], which is an NP-Hard problem. In order to realize the strategy proposed in [28], the strategy has the limitation of requiring a complete copy of all contents in the network. [11] proposes a cache-aware network planning strategy, including location selection of cache nodes and content placement, as well as determination of request forwarding path, and establishes an optimization objective with multiple costs as the main body. However, as one of the important components of optimization problem, the setting of user requests is too simple where one user only initiates requests for one content, and thus the performance gain is limited in the actual system.

In summary, the current arbitrary topology caching algorithm has the problem that global optimization of the caching performance is not achieved due to the limited forwarding range, and the caching algorithm combined with the NRR forwarding strategy is unable to realize a multi-hop fashion. Therefore, we intend to combine the two, that is, to implement a proactive caching placement algorithm in arbitrary topology combined with the multi-hop NRR forwarding, and formulate an optimization problem. In the solution process, we utilize the auction algorithm and the matching algorithm respectively, and introductions of the two theories can be found in [30] and [31].

III. System Model and Problem Formulation

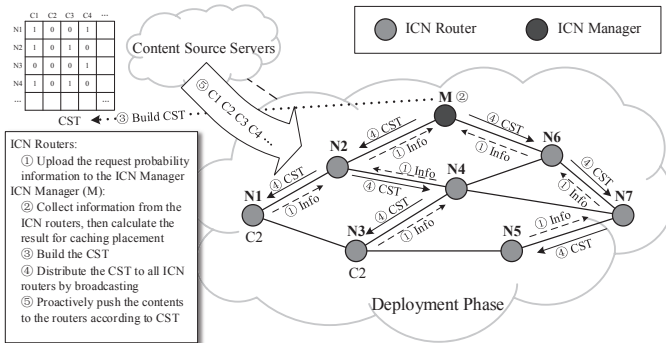
In this paper, we proposed a caching placement algorithm for arbitrary topology. In this section, the network topology, the working process of the proposed algorithm and the content request distribution in our scenario is described, and the formulation of a caching placement problem combined with the multi-hop NRR forwarding is given. For the sake of clarity, we summarize the notation used throughout the paper in Table I.

A. Network Topology

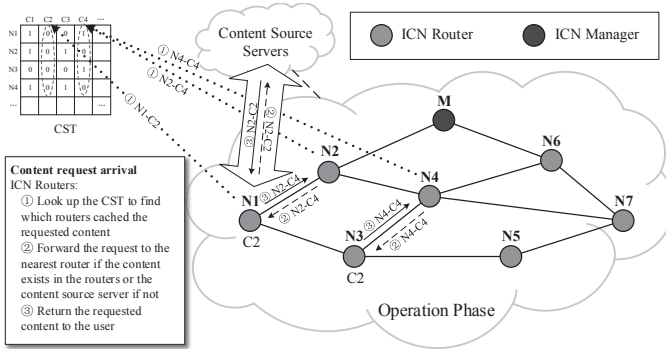
The proposed caching placement algorithm applies to the scenario of a management domain of Internet Service Provider (ISP) or Autonomous System (AS) [32], and the network topology within the domain is arbitrary topology. As shown in Fig. 2, the domain contains ICN routers and an ICN manager node. The ICN routers have the functions of name resolution, content caching, and routing and forwarding. Some ICN routers are connected with multiple user nodes, responsible for collecting and forwarding user-initiated content requests.

TABLE I: Summary of the Notation

Symbol	Meaning
$\mathbf{R}$	Set of cache nodes
$\mathbf{O}$	Set of contents
$\mathbf{U}$	Set of users
J	Number of cache nodes, $J = \|\mathbf{R}\|$
I	Number of contents, $I = \|\mathbf{O}\|$
K	Number of users, $K = \|\mathbf{U}\|$
S_j	Size of cache spaces of router j
p_{ki}	Request probability of user k for content i
p_{ij}	Request probability at node j for content i
d_{kj}	Hop count from user k to node j
d_{ki}	Hop count from user k to the source server of content i
x_{ij}	0-1 caching placement variable: $x_{ij} = 1$ if j caches i.
y_i	Number of replicas of content i



(a) The Deployment Phase



(b) The Operation Phase

Fig. 2: Working process of the proposed caching placement algorithm

The user nodes are not shown in the figure for the sake of simplicity. The ICN manager can communicate with all the ICN routers in the domain, and is responsible for collecting information to make caching placement decisions, and giving the Caching State Table (CST) to routers for forwarding requests. Hereinafter, ICN routers can also be called cache nodes.

B. Working Process of the Proposed Caching Placement Algorithm and the NRR Forwarding Strategy

The working process of the proposed caching placement algorithm can be described in two phases: the deployment phase and the operation phase. The working process in the deployment phase is given in Fig. 2(a). Step 1: all the ICN routers upload the user request probability information to the ICN manager. Step 2: the manager collect the information from all the routers, and calculate the result for the caching placement algorithm utilizing the request probability and the hop count information. Step 3: build the CST according to the result. Step 4: distribute the CST to all the routers by broadcasting. Step 5: proactively push the contents from the content source servers to the routers according to the CST. This paper proposes an active cache placement strategy that pushes specific content to various cache nodes in the network. Every once in a while, the cached content of the entire network will be updated, and the corresponding CST of each node will also be updated. The original cached content in the node is reserved if it appears in the new CST; if it is not in the new CST, it is replaced with the new content. Therefore, the strategy can guarantee to search for the most recent content based on CST. Each time the CST is updated, the original contents in the cache space would also be updated according to the CST. It is noteworthy that the user request probability at each router is obtained independently by recording the requests initiated by users accessing the respective routers and calculating the probability with a certain method. Since this paper mainly focuses on how to use the request probability instead of how to generate it, and the method of generating the probability does not affect the caching algorithm, this paper did not introduce the specific probability generation method.

After the deployment phase, the working process enters the operation phase, which is given in Fig. 2(b). When a content request arrives at a router, the router will follow the following process. Step 1: look up the CST to find which routers cached the content. Step 2: if the lookup result is positive, according to the adopted NRR forwarding strategy, the router forwards the request to the nearest router which cached a replica of the content. If the lookup result is negative, the router forwards the request to the content source server. Here we use some examples to illustrate the process. In Fig. 2(b), user requests for content C4 arrive at router N2 and N4. The two routers both look up the CST and find out router N1 and N3 cached the content. According to the NRR forwarding strategy, N2 and N4 forward the request to N1 and N3 respectively. On the other hand, user request for content C2 arrives at router N1. N1 looks up the CST to find out there are no replica of C2. Then N1 forwards the request to the content source server. Step 3: After receive the requests, router N1, N3 and the content source server return the requested contents to the users along the forwarding path respectively. It is noteworthy that the 'nearest router' here means the router with the smallest hop count distance.

In the following, the method for determining the forwarding destination for users is described. According to the NRR forwarding strategy, user requests will be forwarded to the nearest cache node which cached the replica of the content. Thus, by determining the cache nodes to which the users are forwarded, a plurality of user sets $\mathbf{U}_j^i$ against different cache nodes for content i can be obtained. The detailed steps are as follows.

First, the quantity of user sets and the corresponding cache node number are determined. The quantity of user sets for

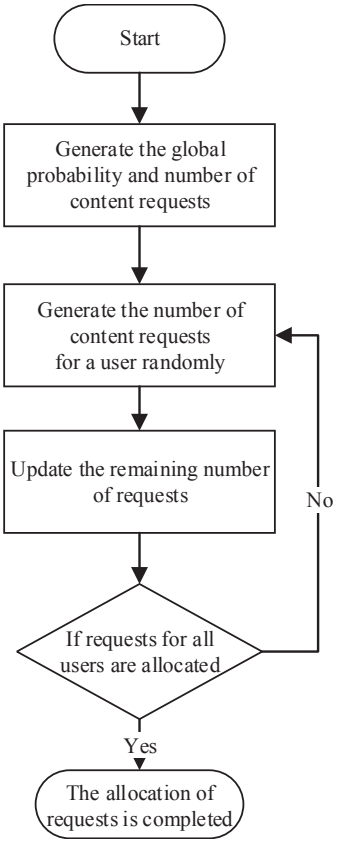


Fig. 3: User Request Generation Process

content i is y_i, the number of replica of content i in the network, where the cache node number set $\mathbf{J}_i$ is defined as

$$\mathbf{J}_i = \{j | x_{ij}{=}1\} \tag{1}$$

Next, the node number j_k^i of the forwarding destination for user k when requesting content i need to be determined, which is defined as

$$j_k^i = \arg\min_j d_{kj}, \quad j \in \mathbf{J}_i \tag{2}$$

The expression means to assign user k to the nearest node that can satisfy the request.

Furthermore, the set of users who are forwarded to node j when requesting the content i is obtained by assembling the users belonging to the same cache node are into a set, which is defined as

$$\mathbf{U}_j^i = \left\{u_k | j_k^i = j\right\} \tag{3}$$

The set of all users $\mathbf{U}$ can be obtained by $\bigcup_{j \in \mathbf{J}_i} \mathbf{U}_j^i = \mathbf{U}$.

C. Content Request Distribution

In caching models of many classical literatures [6], [7], [10], both the network-wide and the single-node content requests follow the Zipf distribution [33], where content i is requested with probability $q_i = c \cdot i^{-\alpha}$, $c = \left(\sum_{i \in \mathbf{O}} i^{-\alpha}\right)^{-1}$. The value of α indicates the concentration degree of content requests. A bigger α means that fewer contents attract the majority of the requests. Under this setting, the difference of user preferences for contents is not reflected, and homogeneity of the user request is serious. Besides, in [8], [11], the content requests at a single node are quite different, which embodies user preferences. However, the network-wide content requests do not follow the Zipf distribution, and it lacks the generality of the network requests following the Zipf distribution.

Different from the above-mentioned request setting methods, in our model, we use a different user request generation process to achieve the differentiated content request distribution at each cache node while keeping the network-wide requests following Zipf distribution. In our process, the user requests for contents are generated randomly under the limitation of the total number of content requests. Fig. 3 gives the user request generation process adopted in our scenario.

First, generate the global probability for all the contents and the total number of requests for all the contents according to the global probability and the total number of user requests. The generated global probability represents the network-wide request probability, which reflects the overall popularity distribution of the contents. Second, generate random request numbers of all the contents for a user under the restrict of the total request numbers. The request probability of each content for a user is obtained by calculating the proportion of the request number of each content in total request number. Third, update the remaining total request number. Then repeat the second and the third steps to generate the request probability for other users until request probability for all the users are generated.

D. Problem Formulation

In this section, we will describe the global optimization model of the caching placement problem combined with the multi-hop NRR forwarding.

In order to reduce user delay and improve load balancing level of cache nodes in the network, we formulate a multi-metric proactive caching placement optimization problem. The first metric is to reduce user delay, and we use user average hop count for content acquisition to , which is related to the user delay directly. The smaller the average hop count, the smaller the user delay. On the other hand, we borrowed the concept "load balancing" in heterogeneous cellular networks to reflect the uniformity of load distribution of cache nodes in the network, and the fairness index of the distribution of the user requests hit on each cache node is utilized as the metric for the load balancing level of the cache nodes. The higher the fairness index, the higher the load balancing level.

The user average hop count for content acquisition is represented by $d(\mathbf{X})$, which is derived by

$$d(\mathbf{X}) = \frac{1}{I \cdot K} \cdot \sum_{i \in \mathbf{O}} d_i(\mathbf{X}) \tag{4}$$

where,

$$d_i(\mathbf{X}) = \begin{cases} \sum\limits_{j \in \mathbf{R}} \sum\limits_{k \in \mathbf{U}_j^i} x_{ij} p_{ki} d_{kj} & \exists x_{ij} > 0, j \in \mathbf{R} \\ \sum\limits_{k \in \mathbf{U}} p_{ki} d_{ki} & \forall x_{ij} = 0, j \in \mathbf{R} \end{cases} \tag{5}$$

(5) consists of two parts, which are hop count consumed when a request is hit or not hit in the in-network cache nodes

respectively. $\mathbf{U}_j^i$ represents the set of users who are forwarded to the node j when requesting the content i, which is described in Section III-B.

On the other hand, we use the equation of Jain's fairness index to calculate the fairness index $r(\mathbf{X})$, which is derived by

$$r(\mathbf{X}) = \frac{\left(\sum_{j=1}^{J} p_j\right)^2}{J \cdot \sum_{j=1}^{J} {p_j}^2} \tag{6}$$

where p_j represents the proportion of user requests to hit at the cache node j, which is defined as

$$p_j = \frac{1}{K} \sum_{i \in \mathbf{O}} \sum_{k \in \mathbf{U}_j^i} x_{ij} p_{ij}, \quad j \in \mathbf{R} \tag{7}$$

Then, we module an optimization problem with an objective of a trade-off between the average hop count and the fairness index, where σ_1 and σ_2 are trade-off parameters between the two metrics. The formula of the optimization objective is given by

$$\min_{\mathbf{X}} \sigma_1 d(\mathbf{X}) - \sigma_2 r(\mathbf{X}) \tag{8}$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{O}} x_{ij} \leq S_j, \quad \forall j \in \mathbf{R}, \tag{8a}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathbf{O}, \quad \forall j \in \mathbf{R}, \tag{8b}$$

(8a) sets the cache space limit for each cache node, where the cache space is measured by the amount of content replicas, and (8b) limits the problem to a 0-1 planning problem. The formula (8) is NP-Hard and can not be solved directly. So we solve the problem in two steps. In the following section, we will describe the algorithm in detail.

IV. Proposed Caching Placement Algorithm

We solve the caching placement problem in two sub-problems, content replica allocation sub-problem and content replica placement sub-problem. First, we reformulate the original problem into a content replica allocation problem by approximating the optimization objective. In the content replica allocation problem, cache spaces are resources and need to be assigned to the contents. Different contents occupy different number of cache spaces, so we utilize the auction algorithm to allocate the cache spaces to all the contents. Second, the replica number of each content is used as a constraint for the content replica placement problem. In the content replica placement problem, the number of replicas for each content is fixed, and hence we only need to determine where to cache them to achieve the best performance. Here we utilize the matching algorithm to find the best matching pairs. In this section, we will describe the two algorithms in detail.

A. The Content Replica Allocation Algorithm Based on Auction Theory

In this section, we want to achieve the exact number of replica for each content y_i. But at first, we need to make a reasonable approximation to the (4) in the optimization objective. When there are no replica of content i in the network, the total hop count required for all users to obtain the content i may be expressed as:

$$d_{\max}^i = \sum_{k \in \mathbf{U}} p_{ki} d_{ki} \tag{9}$$

Next, we need to arrive at an approximation of the total hop count for all users to obtain the content i with y_i replicas cached. Inspired by [34], we propose the hypothesis of the relationship between the hop count for content acquisition by the user and the number of content replica in the network, which can be expressed as $\tilde{d}_i(y_i) = d_i \beta^{y_i}, y_i \geq 0$, where β is the attenuation coefficient, and the value range is $(0, 1]$, and $\tilde{d}_i$ decreases with the increase of y_i. In the literature, the value is set artificially according to the size of the network. Considering the user interest in the contents, the user request distribution is not uniform, and different levels of request probability will reflect in different edge nodes. For example, in some cases, user requests for a certain content are focused on fewer nodes, and fewer replica is required to significantly reduce the average hop count required. In other cases, user requests for a certain content are more dispersed, more replica is needed to achieve the same effect. To solve this problem, we design the formula of β according to the distribution of user requests.

To get the concentration degree of user requests in the network, we utilized the concept of species evenness. Species evenness refers to the distribution of the individual numbers of all species in a certain area, and it reflects the evenness degree of all species. The more the species, the larger the species evenness. When the quantity of species is the same, the closer the individual numbers of the species, the larger the species evenness. In our scenario, a certain content corresponds to a certain area, a certain cache node corresponds to a certain species, and the number of requests at a certain cache node corresponds to the individual numbers of a certain species. For ease of understanding, we renamed species evenness to request evenness. Due to the fixed network topology, the quantity of cache nodes for different contents is the same, so the request evenness reflects the concentration degree of user requests in the network directly. The larger the request evenness, the more concentrated the user requests.

Since Loyd and Pielou et al. proposed the measuring method of evenness, several species evenness have been developed. Currently, Pielou evenness is mainly used, which is based on Simpson index and Shannon-Wiener index. Other evenness indices include Alatalo, Sheldon, Hiep and Hurlbert evenness indices.

According to the objective of this paper, the range of our request evenness should be between 0 and 1, so the Shannon-Wiener index is adopted. The formula for the request evenness

of content i (Pielou evenness) can be expressed by:

$$P_i = \frac{H_i}{H_{\max}} \tag{10}$$

where H_i is the Shannon-Weiner index of content i, and $H_{\max}$ is the theoretical maximum of H_i. Therefore, the Shannon-Weiner index is calculated as follows:

$$H_i = -\sum_{j \in \mathbf{R}} \left(p_{ij} \ln p_{ij}\right) \tag{11}$$

where p_{ij} can be obtained by utilizing the request history of node j, and the method to obtain the value is not introduced here for it is not the focus of this paper. $H_{\max}$ is obtained when all requests are evenly distributed where $p_{ij} = \frac{1}{J}, \forall j \in \mathbf{R}$, the formula is:

$$H_{\max} = -\sum_{j \in \mathbf{R}} \left(p_{ij} \ln p_{ij}\right) = -\sum_{j \in \mathbf{R}} \left(\frac{1}{J} \ln \frac{1}{J}\right) = \ln J \tag{12}$$

In summary, the formula for calculating the request evenness is as follows:

$$P_i = \frac{-\sum\limits_{j \in \mathbf{R}} \left(p_{ij} \ln p_{ij}\right)}{\ln J} \tag{13}$$

The range of values for P_i is $(0, 1]$. When the request is concentrated on a few edge nodes, the calculated value of P_i is close to 0; when requests are scattered across most edge nodes, the calculated value of P_i is close to 1.

Let $\beta = P_i$, and when the replica number of content i is y_i, the formula for approximate hop count $\tilde{d}_i$ is given by:

$$\begin{aligned} &\tilde{d}_i\left(y_i\right) = d_{\max}^i P_i^{y_i} \\ &= \sum_{k \in \mathbf{U}} p_{ki} d_{ki} \left[\left(-\tfrac{1}{\ln J}\right) \sum_{j \in \mathbf{R}} \left(p_{ij} \ln p_{ij}\right)\right]^{y_i} \end{aligned} \tag{14}$$

The approximation of the total hop count to acquire all the contents can be obtained by:

$$\tilde{d}\left(\mathbf{Y}\right) = \sum_{i \in \mathbf{O}} \tilde{d}_i\left(y_i\right) = \sum_{i \in \mathbf{O}} d_{\max}^i P_i^{y_i} \tag{15}$$

where $\mathbf{Y}$ is the vector for the number of replica of all contents, $\mathbf{Y} = \left(y_i\right),\ i \in \mathbf{O}$ and $y_i \in \{0, 1, \ldots\}$.

The original optimization objective can be rewritten as:

$$\min_{\mathbf{Y}} \tilde{d}\left(\mathbf{Y}\right) = \frac{1}{I \cdot K} \cdot \sum_{i \in \mathbf{O}} d_{\max}^i P_i^{y_i} \tag{16}$$

In the following, the exact number of replica of each content in the network y_i is determined to minimize the value of $\tilde{d}\left(\mathbf{X}\right)$ utilizing auction theory. Here one cache space can hold one replica of a content, and the amount of cache space occupied by each content is the number of replica of the content in the network. In order to allocate cache spaces to different contents, this paper uses the auction method to auction the cache space as a commodity to the content as a buyer. By assigning a valuation to each cache space for different contents, every cache space is auctioned to the content with highest valuation until all cache spaces are auctioned. Eventually, different contents will acquire different amounts of cache space. Some contents may acquire one or more than one cache spaces, while other contents may not acquire any spaces.

Algorithm 1 Proposed Auction-Based Content Replica Allocation Strategy

Input: $\mathbf{O}$, $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{d}$
Output: The matrix of number of content replica $\mathbf{Y}$

```
Initialization: t = 1, n_i^t = 0, and S_remain = S.
repeat
    Calculate v_i^t for all i.
    a_t = arg max_i v_i^t
    n_{a_t}^t = n_{a_t}^t + 1
    S_remain = S_remain − 1
    t = t + 1
until S_remain = 0
y_i = n_i^t for all i
```

The valuation on a cache space for a content can be derived from the previous approximated optimization objective function. Since $r\left(\mathbf{X}\right)$ for a single content can not be achieved, only $d\left(\mathbf{X}\right)$ is considered in the valuation function. The valuation is specific to a certain cache space, so the hop count of each content in different number of replica is required. The hop count decreases as the number of replica increases, so the hop count reduction is adopted as the valuation. The formula for the valuation of content i at the t-th auction is as follows:

$$\begin{aligned} &v_i^t = \tilde{d}_i\left(n_i^t\right) - \tilde{d}_i\left(n_i^t + 1\right) = d_{max}^i P_i^{n_i^t - 1} - d_{max}^i P_i^{n_i^t + 1} \\ &= \sum_{1 \le k \le K} p_{ki} d_{ki} \left[-\left(\ln J\right)^{-1} \sum_{1 \le j \le J} p_{ij} \ln p_{ij}\right]^{n_i^t} \\ &- \sum_{1 \le k \le K} p_{ki} d_{ki} \left[-\left(\ln J\right)^{-1} \sum_{1 \le j \le J} p_{ij} \ln p_{ij}\right]^{n_i^t + 1} \end{aligned} \tag{17}$$

where n_i^t represents the number of cache space already acquired by content i at the t-th auction.The value is 0 in an initial state, i.e, $n_i^1 = 0$. At the end of the whole auction process, the final value of n_i^t can be abbreviated as n_i.

The result of this auction algorithm can obtain the minimum value of the total hop count, and we will prove that.

Lemma 1: The result of the auction can obtain the minimum of the total hop count.

Proof 1: See Appendix A .

In the following, the auction process is described below:

1) Initialize the value of remaining cache space $S_{remain} = S$, the number of cache space acquired by each content $n_i = 0$, and the sequence number of the auction $t = 1$.
2) The t-th auction begins. calculate the valuation v_i^t of the t-th cache space being auctioned for each content.
3) The cache space is auctioned to the content a_t having maximal valuation. The number of auctioned cache space $n_{a_t}^t$ corresponding to content a_t is increased by 1, and the remaining cache space S_{remain} is decreased by 1. The t-th auction ends, and the sequence number t is increased by 1.
4) Repeat steps 1) and 2) until the condition is met: $S_{remain} = 0$, then the whole auction ends.

After the auction process, the winner of each auction a_n, $1 \leq n \leq S$ can be obtained, and the number of cache space acquired by each content is y_i. The algorithm 1 gives the pseudo-code for the proposed auction-based content replica allocation algorithm. In the algorithm, $\mathbf{Q}$ is the user request matrix, and $\mathbf{d}$ is the hop count matrix between the users and nodes. At the end of the auction, the obtained $\mathbf{Y}$ is the vector of the number of replica for each content. Next, we will use y_i to obtain the value of x_{ij}, that is, to determine the location of each replica.

B. The Content Replica Placement Algorithm Based on Matching Theory

In the previous auction algorithm, the number of replica of each content in the network has been obtained, and then the location of each copy of content needs to be obtained. The problem is the optimization problem of the relationship between contents and cache nodes. The equation of objective of the optimization problem remains the same as equation (8), and the limitation of the number of replica is added to the constraints of the optimization problem based on the original constraints:

$$\sum_{j \in \mathbf{R}} x_{ij} = y_i \ \ \forall i \in \mathbf{O} \tag{18}$$

A replica of a content is cached on a cache node, which is equivalent to a matching relationship between the content and the cache node; all replicas of all the contents are cached on the cache nodes all over the network, which is equivalent to a many-to-many matching problem between contents and cache nodes. In this matching, a single content can be matched to multiple cache nodes, and a single cache node can be matched to multiple contents. Based on the above descriptions, we give the following definitions.

Definition 1: A many-to-many matching μ is a function from the set $\mathbf{O} \cup \mathbf{R}$ into the set of unordered families of elements of $\mathbf{O} \cup \mathbf{R} \cup \{0\}$ such that:

1) $\mu(i) = \mathbf{R}_\mu^i \subseteq \mathbf{R}$;
2) $\mu(j) = \mathbf{O}_\mu^j \subseteq \mathbf{O}$;
3) $|\mu(i)| = y_i, \ \ \forall i \in \mathbf{O}$
4) $|\mu(j)| \leq S_j, \ \ \forall j \in \mathbf{R}$
5) if and only if $\mu(j) \in \mathbf{O}$, $\mu(i) \in \mathbf{R}$;
6) $i \in \mu(j) \Leftrightarrow j \in \mu(i)$.

In 1) and 2), the symbol μ has different meanings for different parameters. When the parameter is content i, $\mu(i)$ represents an unordered subset of the set of cache nodes matching the content i; When the parameter is a cache node j, $\mu(j)$ represents an unordered subset $\mathbf{O}_\mu^j$ of the set of contents $\mathbf{O}$ matching the cache node j. 3) and 4) represent quotas of each matching, where y_i is the number of replica of each content obtained in the previous process, and S_j is the cache space limit for each cache node. 5) indicates that the matching result of i or j must be a subset of the set $\mathbf{R}$ or $\mathbf{O}$. 6) indicates that the matching parties are mutual.

Definition 2: There doesn't exist that one content is unacceptable to any cache node nor one cache node is unacceptable to any content, and a player in the matching must be individually rational.

The matching in this paper contains externalities, which is reflected in that when existing more than one replica of a content, and the location of one replica will affect the utility of the other replicas. The reason is that the addition or reduction of replica and the change in the placement of replicas would cause a change to the user set, and consequently changes the utility value of the content i. In many-to-many matching with externalities, a stability concept cannot be defined straightforwardly because the gain from a matching pair depends on the matching results of other players. Inspired by the definition of exchange stability, a swap matching is defined to achieve stability [31]. Specifically, an swap matching is defined as $\mu_i^{i'} = \{\mu \backslash \{(i,j),(i',j')\} \cup \{(i',j),(i,j')\}\}$, where $j \in \mu(i)$, $j' \in \mu(i')$, i.e., the cache node that matches with the content i changes from j to j', and meanwhile the cache node that matches with content i' changes from j to j'. Based on the operations of swap matching, we will give the concept of swap-blocking pairs as follows, and then introduce the two-sided exchange stability.

Definition 3: When there two players i and i' , such that $d\left(\mu_i^{i'}\right) < d(\mu)$, the (i,i') are called an swap-blocking pair in the matching μ.

$d(\mu)$ represents the overall utility value of the matching μ. The features of swap-blocking pairs ensures that if a swap matching is approved, the value of the overall utility related to hop count is decreased. In this paper, both of the contents and the cache nodes are of no selfishness, so as long as the overall utility can be decreased, the pair of matching players is a swap-blocking pair, without satisfying the requirements that their respective utility will not be increased.

Definition 4: The matching μ are two-sided exchange-stable if and only if there are no swap-blocking pairs.

This definition gives a necessary and sufficient condition for judging the stability of many-to-many matching problems with externalities. Below we will prove that there is always a stable matching in the model considered in this paper. Before the proof is given, a local minimum needs to be defined.

Definition 5: If there are no swap-blocking pairs such that $d(\mu') < d(\mu)$, $d(\mu)$ can be a local minimum.

Lemma 2: All local minima of d are two-sided exchange-stable.

Proof: assume the matching μ is a local minimum, then according to definition 5, there is no swap-blocking pairs in the matching μ. And then according to definition 4, it is proved that the matching μ is two-sided exchange-stable. The proof is completed.

In the model considered in this paper, because the number of matching is finite, there must be local minima, so it is also proved that there is always two-sided exchange-stable matchings in the model.

Next we introduce the definition of the overall utility $d(\mu)$. Since the purpose of the matching process is to find the solution of the optimization objective and the matching has the externalities, the utility in the swap matching process is consistent with the optimization objective in this paper, that is, the overall utility of the matching, which can be expressed as:

$$d(\mu) = \sigma_1 d(\mathbf{X}) - \sigma_2 r(\mathbf{X}) \tag{19}$$

The value of overall utility is used to find better matchings in the swap matching process, and the lower the utility value, the better the performance. For example, assuming that μ_1 and μ_1 are two different matchings, we have the following relationship:

$$\mu_1 \succ \mu_2 \Leftrightarrow d\left(\mu_1\right) < d\left(\mu_2\right) \tag{20}$$

where $\succ$ is the preference relation, indicating that matching μ_1 is superior to matching μ_2 only when the value of the overall utility of matching μ_1 is smaller than matching μ_2.

Next, in order to build the initial matching state between the contents and the cache nodes, we need to build a preference list between the content and the cache node. Each content has different preferences for different cache nodes, and each cache node has different preferences for different contents similarly. Here these two kinds of preference values are defined.

First, based on the optimization objective, we define the preference value of a content over a caching node. Since $r\left(\mathbf{X}\right)$ for a single content can not be achieved, only $d\left(\mathbf{X}\right)$ is considered in the preference value. Furthermore, the preference value is the initial value of the content, it is assumed that there is only content i in the network, and changing the matching cache node j does not affect the value of the cache miss part in the formula of the optimization objective. So the hop count caused by other contents is not considered, only the hop count required is calculated when only one replica of content i is cached on cache node j, which can be expressed as:

$$d_i\left(j\right) = \sum_{k \in \mathbf{U}} p_{ki} d_{kj} \tag{21}$$

Assuming that for a given i and any two different cache nodes j and j', we have the following relationship:

$$j \succ_i j' \Leftrightarrow d_i\left(j\right) < d_i\left(j'\right) \tag{22}$$

which indicates that the content i prefers j than j', only when the utility value for content i is smaller on j than j'.

Similarly, we define the preference value of a cache node over a content. Since it is the initial preference of the cache node, it is assumed that other cache nodes do not exist, but changing the matching content i will affect the value of the cache miss part in the formula of the optimization objective, so the hop count required for acquiring all the contents is considered in the preference value of a cache node, which can be expressed as:

$$d_j\left(i\right) = \sum_{k \in \mathbf{U}} p_{ki} d_{kj} + \sum_{i' \in \{\mathbf{O} \setminus i\}} \sum_{k \in \mathbf{U}} p_{ki'} d_{ki'} \tag{23}$$

Assuming that for a given cache node j and any two different contents i and i', we have the following relationship:

$$i \succ_j i' \Leftrightarrow d_j\left(i\right) < d_j\left(i'\right) \tag{24}$$

which indicates that the cache node prefers the content i to i' only when the utility value for cache node j is smaller on i than i'.

Now the initial matching state between the contents and the cache node can be obtained utilizing the extended deferred acceptance (EDA) algorithm based on the preference list established above, where we assume that the content proposes to cache nodes. In the EDA-based initialization process, first, the contents and cache nodes construct their own preference lists based on 22 and 24 respectively. Then each content proposes to the most preferred cache node in its preference list. At the acceptance phase of cache nodes, each cache node accepts the content with prior preferences and rejects the others. The initialization process ends when all contents are matched to the cache node or every unmatched contents has been rejected by every cache node.

Algorithm 2 Proposed Matching-Based Content Placement Strategy

Input: $\mathbf{O}$, $\mathbf{R}$, $\mathbf{Y}$, $\mathbf{Q}$, $\mathbf{d}$
Output: The stable matching μ^* and the replica placement matrix $\mathbf{X}$.

1: Initialization: Construct the initial content-caching node matching μ.
2: **repeat**
3: For any content, select one caching node $j \in \mathbf{R}_\mu^i$ randomly. It searches for another content $i' \neq i$ and one matching node $j' \in \mathbf{R}_\mu^{i'}$.
4: **if** $d\left(\mu_i^{i'}\right) < d\left(\mu\right)$ **then**
5: The caching nodes agree the swap operation.
6: Update the matching μ.
7: **until** Reach the maximum iteration
8: $\mu^* = \mu$

Algorithm 2 presents the process for solving the problem of matching between contents and cache nodes. First, the initial matching state is obtained by the initialization process, then the contents and the matching nodes are randomly selected for the swap operation, and if the utility value is smaller, the swap operation is carried out. Repeat the swap operation until reaching the predefined maximum number of iterations. The approach of ending the iteration when a stable matching is reached is not adopted here, because in many-to-many matching problems with externalities, although there is always two-sided exchange-stable matchings, due to the high order of magnitude of each parameter and the update to the utility value after each swap matching, it will take a lot of time to find a stable matching. The experimental results show that the utility value tends to be stable after a certain number of swap operations. Therefore, in this algorithm, we take the approach of ending the iteration until reaching the predefined maximum number of iterations.

In Algorithm 2, $\mathbf{X}$ is a content placement matrix that corresponds to a stable matching μ^*, and the values of the elements in the matrix can be obtained by:

$$x_{ij} = \begin{cases} 1 & j \in \mu^*\left(i\right) \\ 0 & j \notin \mu^*\left(i\right) \end{cases} \tag{25}$$

Up to now, we find the solution of the optimization problem by matching algorithm, and obtain the scheme of content placement algorithm, where a replica of the content i is cached at the cache node j when $x_{ij} = 1$, otherwise it will not be cached.

C. Analysis of the Proposed Algorithms

1) Complexity: The proposed two-step caching algorithm is based on auction algorithm and matching algorithm. In the following, we will analyze the computational complexity of these two algorithms. In Algorithm 1, evaluations of I contents are calculated in each loop, and there are total S loops. The complexity is $O(IS)$. In the initialization procedure of algorithm 2, each content propose to cache nodes, and then each cache node decides which content to accept based on its preference list. In the worst case the proposing number of a content is $J - y_i$. The complexity is $O\left(I^2J^2\right)$. In the swap procedure of algorithm 2, there are I contents at J cache nodes can perform swap operations. Each content matches with y_i cache nodes, and each cache node matches with S_j contents. Therefore, the maximum swap operation number for content i is $y_i(J - y_i)$ in each iteration, and the maximum swap operation number for I contents is $\frac{1}{2}IS_jy_i(J - y_i)$. Given a number of total iterations V, the complexity is approximated as $O(VIJS_jy_i)$. As a result, the complexity of the entire caching placement algorithm can be calculated as $O\left(I^2J^2 + IS + VIJS_jy_i\right)$.
2) Stability and convergence: In the auction procedure of algorithm 1, one content is selected to allocate one cache space in each loop, and the loop process ends with the completion of the cache space allocation. After the swap process of algorithm 2, a two-sided exchange-stable matching is formed between the contents and the cache nodes. Because the utility value decreases monotonously in the swap process and there is a floor limit on the utility value due to the limited cache space. After a finite number of swap operations, algorithm 2 will get a local solution, that is, converge to a two-sided exchange-stable status.

V. Simulation Results

In this section, in order to verify the performance of the proposed caching algorithm, we simulate the proposed content placement algorithm and compare it with other algorithms. In the simulation, the Matlab simulation platform is used to implement the caching placement algorithm. Currently, there is no existing caching placement algorithm that perfectly matches our proposed scenario. Therefore, we chose two typical ICN caching algorithms to compare with the proposed caching placement algorithm for arbitrary topology (CPAT) in the simulation: 1) MPC algorithm [35], a caching placement algorithm based on local popularity; 2) RND algorithm [36], a probabilistic caching placement algorithm. All the caching placement algorithms adopts NRR algorithm as the request forwarding algorithm in the simulation, and the specific forwarding method has been described in detail before, so we will not repeat it here.

A. Simulation Assumption

The topology used in this paper is arbitrary topology. Without loss of generality, we use randomly generated arbitrary topology as the simulation topology. The specific generation method is: First, the nodes are randomly scattered in the plane, and then the connection is randomly established according to a certain probability between all the nodes. In our simulation, the probability is 10%. Second, the gateway node and the cache nodes are randomly selected from the nodes in the network, and several user nodes are connected to each cache node.

The default parameters used in the simulation are given in Table II. For convenience, it is assumed that each cache node has the same size of cache spaces and that each cache space can accommodate one content. Two Zipf distribution parameters have been considered in the simulation: $\alpha = 1.0$ and $\alpha = 0.7$. $\alpha = 1.0$ is used to model a relatively skewed popularity distribution where few contents are frequently requested, whereas $\alpha = 0.7$ better represents less skewed content requests.

TABLE II: Simulation Parameters

Parameters	Value
Number of cache nodes (J)	20
Number of users (K)	200
Number of contents (I)	300
Number of requests by a user (Q_k)	1000
Size of cache spaces for a cache node (S_j)	20
Content Distribution	Zipf (α = 0.7 or 1)

In the simulation, we mainly use the following metrics to evaluate the performance of the caching algorithms.

1) User average hop count for acquiring contents, abbreviated as average hop count, can directly reflect user delay. The smaller the average hop count, the better the performance of the algorithm.
2) Fairness index, can directly reflect the load balancing level of the cache nodes in the network. The higher the value, the more uniform the distribution of requests hitting at the nodes.
3) Utility, is the utility value of the caching placement result, which is recorded in the calculation process. In our simulation, $\sigma_1 = \sigma_2 = 1$, and the values can be adjusted according to the level of average hop count to prevent the gap of the two metrics from being too large.

B. Convergence of the Replica Placement Algorithm

In the process of solving the caching placement algorithm, this paper introduces a matching problem with externalities, and uses swap matching to search the approximate optimal solution. With the proceeding of the swap matching process, the utility value of the matching decreases gradually and approaches to the optimal value. Therefore, in this section, we verify the convergence of the matching algorithm.

As shown in Fig. 4, the four curves in the figure give records of the variation of the utility with the number of swap matchings under different parameter settings. There are 10 cache nodes in the network, each cache node has 10 users connected to it, each user initiates 1000 content requests, and each cache node has 10 cache spaces. The Zipf parameter and the total number of contents are different. Each curve in the graph represents the average value of 20 simulations. First

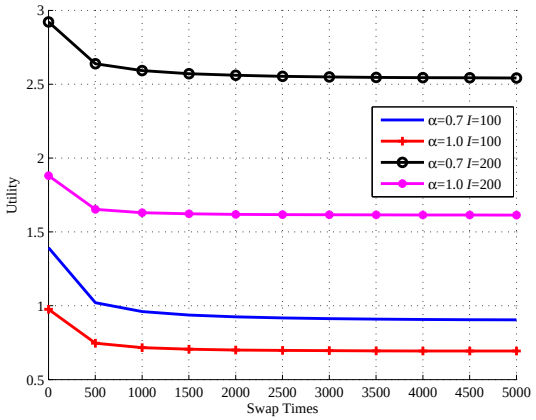


Fig. 4: Records of the utility value for the matching algorithm

observe the overall trend, with the progress of swap operations, the utility values of all the four curves are decreasing. At the beginning of the swap process, the decreasing trend of utility value is relatively fast. After about 300 times of swap operations, the decreasing trend slows down obviously, and after about 1500 times of swap operations, the utility value reaches a relatively stable status. Then observe the trend contrast between the curves, we can see the influences of the popularity parameter and the number of contents. The greater the popularity parameter, the higher the centralization of user requests, the more requests the users generate for a smaller amount of popular contents, the more user requests the cache nodes can accommodate, and therefore the lower the average hop count for the users to acquire the contents. On the other hand, the larger the number of contents, the more dispersed the user requests. In the case of the same cache spaces, the cached contents are unchanged, but the number of user requests that can be satisfied is also reduced accordingly, so the average hop count for the users to acquire the contents is higher.

C. Impact of Network Scale

In this section, we verify the impact of the network scale and the popularity parameters on the performance of the caching algorithms, and here we use the number of cache nodes to reflect the scale of the network. The number of swap operations of the matching algorithm is set to 1000, and each curve in the graph is the average value of 20 simulations. If not indicated, the parameter settings are the same in subsequent simulations.

The network performance comparison with varying cache node number is shown in Fig. 5. The average hop count and the fairness index of the proposed algorithm all performs the best. The utility is in accordance with the result of average hop count and the fairness index, which proved the correctness of the proposed algorithm. From the Fig. 5(a) we can see that with the increase of the number of cache nodes, the average hop count of all strategies is decreasing. This is because the number of cache nodes increases and the total size of cache spaces in the network increases correspondingly, but the total amount of contents does not increase, which allows more contents to be cached in the network. When the hop count reaches a certain level, the downward trend slows down and tends to be stable. As a contrast, the hop count curves of MPC and RND algorithms basically show a linear downward trend. This is because the cached contents for both algorithms do not change with the increase in the number of cache nodes, it is only related to the local popularity of the contents. As the number of cache nodes increases, the number of users also increases proportionally, which leads to more diverse user requests, and the number of cached replicas in the network is also increasing, so the average hop count decreases gradually with the increasing of the number of cache nodes.

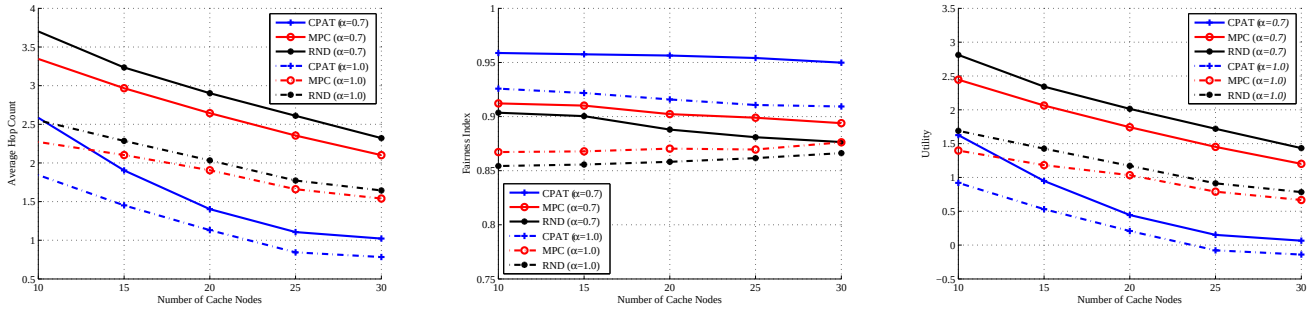
It is noteworthy that the proposed algorithm has an obvious difference in the trend of hop count decline when the popularity parameter is 0.7 and 1.0 respectively, where the curve with the parameter 0.7 is more skewed than the curve with 1.0. This is because when the popularity parameter is smaller, content requests are more decentralized across the network, and caching relatively less popular contents will bring more gain to the caching performance. However, when the size of cache space reaches a certain number, the cache nodes in the network already cover most of the contents requested by the users, and the gain of hop count reduction caused by increasing the number of replicas exceeds the gain caused by caching uncached contents. Therefore, when the cache spaces are saturated, the proposed algorithm with higher popularity tends to cache more existing contents to decrease the average hop count.

From the Fig. 5(b) we can see that with the increase of the number of cache nodes, the curves of the three algorithms show different trends, and the proposed algorithm performs the best. The values of the three algorithms all change within a small interval, and this is the reason why the trends of the curves in Fig. 5(c) are similar to Fig. 5(a). The proposed algorithm is superior under different number of cache nodes, for the algorithm has fully utilized the cache spaces within the network.

D. Impact of Size of Cache Spaces

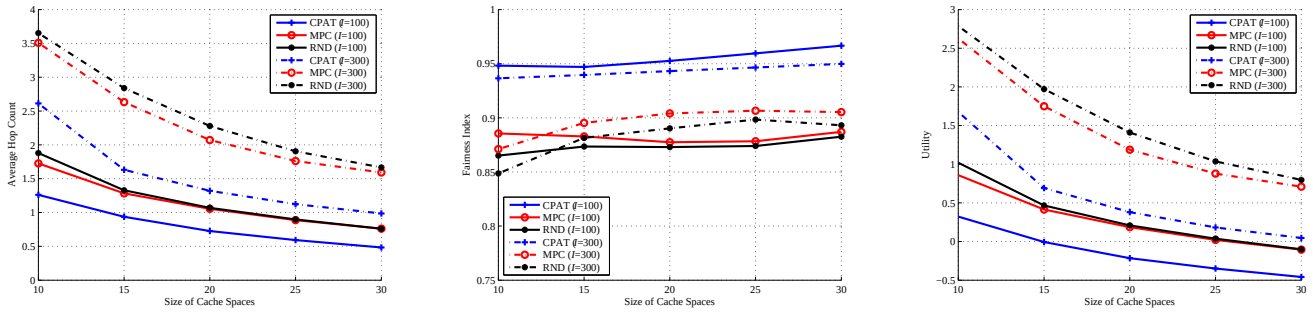
As a constraint of caching algorithm, cache space is one of the important factors that restrict caching performance. In this section, we verify the impact of the size of cache spaces and the number of contents on caching performance.

The network performance comparison with varying size of cache spaces is shown in Fig. 6. The average hop count and the fairness index of the proposed algorithm all performs the best. From the Fig. 6(a) we can see that the average hop count of all algorithms decreases as the size of cache spaces increases. This is because with the increase of the size, the cache nodes can accommodate more contents, which can satisfy more user requests, and make the user requests satisfied at nearer nodes and thus reduce the average hop count. It also can be seen that when the number of contents is bigger, the average hop count is bigger. This is because more contents can not be cached in the network, and more requests need to be satisfied outside the network. From the Fig. 6(b) we can see that with the increase of the size of cache spaces, the curves of the three algorithms



(a) Average hop count (b) Fairness index (c) Utility

Fig. 5: Network performance comparison with varying cache node number



(a) Average hop count (b) Fairness index (c) Utility

Fig. 6: Network performance comparison with varying size of cache spaces

show different trends, and the proposed algorithm performs the best. The values of the three algorithms all change within a small interval, and this is the reason why the trends of the curves in Fig. 6(c) are similar to Fig. 6(a). For the same reason under the scenario of different network scale, the algorithm is also superior under different size of cache spaces.

E. Impact of the Content Number

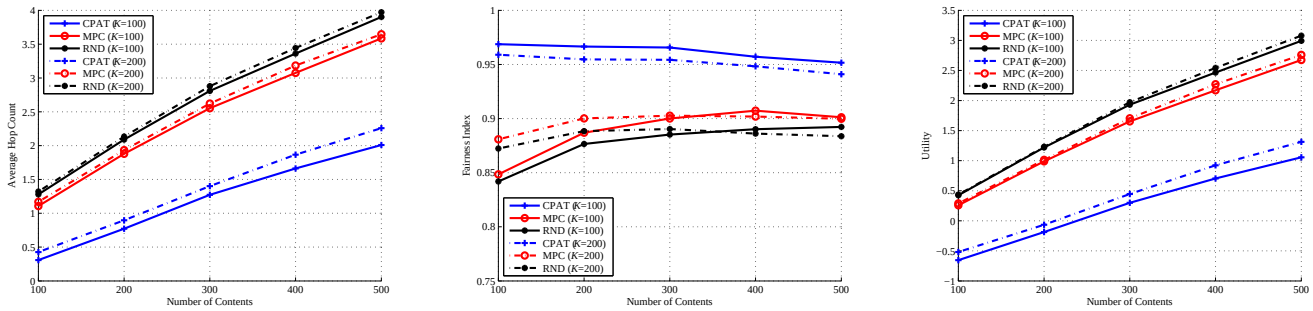
The content number represents the total amount of contents that can be requested by users, which involves elements such as the number of requests for a single content and the distribution of user requests. Therefore, in this section, we verify the impact of the number of contents and the number of users on the performance of the caching algorithms.

The network performance comparison with varying content number is shown in Fig. 7. The average hop count and the fairness index of the proposed algorithm all performs the best. From the Fig. 7(a) we can see that the average hop count of all the algorithms increases as the number of contents increases. With the increasing of the number of contents, the proportion of contents that can be cached is relatively decreased without changing the size of cache spaces, and the proportion of user requests that can be satisfied in the cache nodes are also decreased. Therefore, with the increase of the number of contents, the average hop count for acquiring the contents also increases. We can see that the curves with 100 users performs better than 200 users. This is because according to the generation method of user requests, when the number of users connected to one cache node is bigger, the distributions of content requests at various cache nodes tend to be identical. If the requests for a content are uniformly distributed across all the cache nodes, more replicas are needed to reach the same level of average hop count comparing to the situation that the requests for a content are highly centralized in only a few nodes. And consequently, more cache spaces are needed to reach the same level of cache hit ratio. Therefore, the average hop count of 100 users is lower than the average hop count of 200 users. This also verified the impact of the distribution of content requests on the proposed algorithm.

From the Fig. 7(b) we can see that with the increase of the size of cache spaces, the curves of the three algorithms show different trends, and the proposed algorithm performs the best. The values of the three algorithms all change within a small interval, and this is the reason why the trends of the curves in Fig. 7(c) are similar to Fig. 7(a). The algorithm is also superior under different number of contents and different number of users, because the algorithm has fully considered the request distribution and the distances between users and nodes.

VI. Conclusion

In this paper, we proposed a proactive caching placement algorithm for arbitrary topology, where the content popularity, the distribution of user requests and the distances between users and nodes are considered. The proposed algorithm



(a) Average hop count (b) Fairness index (c) Utility

Fig. 7: Network performance comparison with varying content number

is combined with the multi-hop NRR forwarding strategy, where the caching placement algorithm is not limited by the network topology and can work under any kind of network topologies including arbitrary topology. In comparison to the classic caching placement algorithms, the average hop count for content acquisition is greatly reduced, and the level of load balancing is also improved. Due to the characteristics of the proactive caching placement algorithm, our proposed algorithm is more suitable for network scenarios where the user requests are not highly dynamic and higher network performance is required including user delay and cache hit ratio, and the algorithm will show more superior performance in arbitrary network. The implementation of the proposed algorithm is relatively simple, and the computational complexity is manageable.

APPENDIX A
PROOF OF LEMMA 1

Assuming that the result obtained by the auction is a_t, $t = \{1, 2, \ldots, S\}$, the cache space finally acquired by each content i is n_i, and the sum of the results is S, i.e., $\sum_{1 \le i \le I} n_i = S$.

Since in each auction t, the maximum of v_i^t is selected, the sum of the results of the S auctions $\sum_{1 \le t \le s} v_{a_t}^t$ is also the maximum of $\sum_{1 \le t \le s} v_i^t$, so we have:

$$\max \sum_{1 \le t \le S} v_i^t = \sum_{1 \le t \le S} v_{a_t}^t \tag{26}$$

These S auctions can be grouped by contents, and the amount of auctions is n_i whose auction result is content i. For the ease of writing, the valuation of the result $v_{a_t}^t$ in each auction is redefined as a variable v_i^n with respect to the content i and the cache space n, and the equation of is defined as:

$$v_i^n = \begin{cases} 0 & n = 0 \\ \tilde{d}_i(n-1) - \tilde{d}_i(n) & n > 0 \end{cases} \tag{27}$$

When some contents have not acquired any cache spaces, that is, $n_i = 0$, the value of v_i^0 is $v_i^0 = \tilde{d}_i(0) - \tilde{d}_i(0) = 0$.

Then, re-sum the S auctions according to the content:

$$\sum_{1 \le t \le S} v_{a_t}^t = \sum_{1 \le i \le I} \sum_{1 \le n \le n_i} v_i^n \tag{28}$$

Bring the formula (27) and (28) into (26), we can obtain:

$$\max \sum_{1 \le t \le S} v_i^t = \sum_{1 \le i \le I} \tilde{d}_i(0) - \sum_{1 \le i \le I} \tilde{d}_i(n_i), \quad n_i \ge 0 \tag{29}$$

where $\sum_{1 \le i \le I} \tilde{d}_i(0)$ is a fixed value, so $\sum_{1 \le i \le I} \tilde{d}_i(n_i)$ is the minimum of the total hop count. Because n_i is the auction result for each content, it is proved that the result of auction can get the minimum of the total hop count.

The proof is completed.

REFERENCES

[1] Cisco, "Cisco visual networking index: Forecast and methodology 2017–2022." Report, 2018.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, Conference Proceedings, pp. 1–12.

[3] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2847–2886, 2016.

[4] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.

[5] S. Shan, C. Feng, T. Zhang, and Y. Liu, "A user interest preferences based on-path caching strategy in named data networking," in *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, 2017, Conference Proceedings, pp. 1–6.

[6] J. Li, X. Wang, and M. Huang, "User preferences and data popularity based cache management scheme in information centric networking," *Journal of Chinese Computer Systems*, vol. 36, no. 5, pp. 916–921, 2015.

[7] H. Yan, D. Gao, W. Su, C. H. Foh, H. Zhang, and A. V. Vasilakos, "Caching strategy based on hierarchical cluster for named data networking," *IEEE Access*, 2017.

[8] L. Zhou, T. Zhang, X. Xu, Z. Zeng, and Y. Li, "Generalized dominating set based cooperative caching for content centric ad hoc networks," in *IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2015, Conference Proceedings.

[9] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "Catt: Potential based routing with content caching for ICN," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, Conference Proceedings, pp. 49–54.

[10] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*. ACM, 2013, Conference Proceedings, pp. 27–32.

[11] M. Mangili, F. Martignon, and A. Capone, "Optimal design of information centric networks," *Computer Networks*, vol. 91, pp. 638–653, 2015.

[12] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: Hamlet-like doubts in icn," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. 2342491: ACM, 2012, Conference Proceedings, pp. 7–12.

[13] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *ACM SIGCOMM Computer Communication Review*, vol. 43. ACM, 2013, Conference Proceedings, pp. 147–158.

[14] G. Carofiglio, L. Mekinda, and L. Muscariello, "Joint forwarding and caching with latency awareness in information-centric networking," *Computer Networks*, vol. 110, pp. 133–153, 2016.

[15] T. Zhang, X. Xu, Le Zhou, X. Jiang, and J. Loo, "Cache space efficient caching scheme for content-centric mobile ad hoc networks," *IEEE Systems Journal*, vol. 13, no. 1, pp. 530–541, March 2019.

[16] T. Zhang, H. Fan, J. Loo, and D. Liu, "User preference aware caching deployment for device-to-device caching networks," *IEEE Systems Journal*, vol. 13, no. 1, pp. 226–237, March 2019.

[17] F. Zhang, Y. Zhang, and D. Raychaudhuri, "Edge caching and nearest replica routing in information-centric networking," in *Sarnoff Symposium, 2016 IEEE 37th*. IEEE, 2016, Conference Proceedings, pp. 181–186.

[18] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*. ACM, 2014, Conference Proceedings, pp. 127–136.

[19] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, Conference Proceedings, pp. 316–321.

[20] J. M. Wang and B. Bensaou, "Progressive caching in CCN," in *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012, Conference Proceedings, pp. 2727–2732.

[21] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758–770, 2013.

[22] G. de Melo Baptista Domingues, E. A. de Souza e Silva, R. M. M. Leão, and D. S. Menasché, "Enabling information centric networks through opportunistic search, routing and caching," *CoRR*, vol. abs/1310.8258, 2013.

[23] Y. Zhu, M. Chen, and A. Nakao, "Conic: Content-oriented network with indexed caching," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, March 2010, pp. 1–6.

[24] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, Conference Proceedings, pp. 936–944.

[25] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 864–874, 2014.

[26] A. Araldo, M. Mangili, F. Martignon, and D. Rossi, "Cost-aware caching: Optimizing cache provisioning and object placement in ICN," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, Conference Proceedings, pp. 1108–1113.

[27] K. Naveen, L. Massoulie, E. Baccelli, A. C. Viana, and D. Towsley, "On the interaction between content caching and request assignment in cellular cache networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 2785975: ACM, 2015, Conference Proceedings, pp. 37–42.

[28] H. Xie, S. Guangyu, and W. Pengwei, "TECC: Towards collaborative in-network caching guided by traffic engineering," in *2012 Proceedings IEEE INFOCOM*, 2012, Conference Proceedings, pp. 2546–2550.

[29] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

[30] Y. Zhang, C. Lee, D. Niyato, and P. Wang, "Auction approaches for resource allocation in wireless systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1020–1041, Third 2013.

[31] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer effects and stability in matching markets," *Lecture Notes in Computer Science*, vol. 6982, pp. 117–129, 2011.

[32] S. Shailendra, B. Panigrahi, S. Sengottuvelan, H. K. Rath, and A. Simha, "Distributed optimal caching for information centric networking (ICN)," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on*. IEEE, 2016, Conference Proceedings, pp. 1–6.

[33] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, Conference Proceedings, pp. 126–134.

[34] M. Yang and Z. Fei, "A model for replica placement in content distribution networks for multimedia applications," in *IEEE International Conference on Communications*, 2003.

[35] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *2013 IEEE International Conference on Communications (ICC)*, 2013, Conference Proceedings, pp. 3619–3623.

[36] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proceedings of the Re-Architecting the Internet Workshop*. ACM, 2010, Conference Proceedings, p. 5.